Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 70 (2015) 715 – 722

Procedia Computer Science

4th International Conference on Eco-friendly Computing and Communication Systems

Multi Core Aware Java Library for Computing Protein Stability Indices – A Performance Improvement and Power Consumption Study

Subrata Sinha[a*], Partha Protim Nath, G.C Hazarika[b]

[a]*Centre for Bioinformatics Studies, Dibrugarh University,Dibrugarh-786004,India*
[b]*Department of Mathematics, Dibrugarh University,Dibrugarh-786004,India*

Abstract

Data deluge in post genomic era has necessitated the development of multi-core aware software and libraries. Colossal analysis in the field of bioinformatics demands next generation bioinformatics applications/libraries which ought to be faster and energy efficient. In this work we have developed a multi-core aware java library for peptide properties calculations which are used in various active site prediction, protein stability calculations, protein ligand interactions. The library is capable to detect number of CPU cores at run time and divide the tasks accordingly. Processing time and power consumption has been measured for all methods in the developed library and Bio Java for various granularity of tasks (0.1 MB to 51.2 MB) and the results were compared. The developed library named as pBio shows improved performance upto 37 times than corresponding modules of Bio Java, and up to 6 times more energy efficient approximately than corresponding modules of Bio Java.

Keywords: Multi Core Aware Library, Performance Enhancement, Energy Efficiency.

* Corresponding author. Tel.: +91-848-646-8313
E-mail address: subratasinha@dibru.ac.in

Peer-review under responsibility of the Organizing Committee of ICECCS 2015
doi:10.1016/j.procs.2015.10.109

1. Introduction

Multi-core Computing is one of the HPC model well known for its scalability as its amount of performance increase proportional to the number of cores (Held, J. *et al*. 2006). Multi-core programming helps to implement parallelism which is one of the biggest breakthroughs of performance. Performance can be viewed, in a nutshell, as how fast it takes to complete job (John Darlinton *et al.*1996). Parallelism allows us to increase this performance, because it allows instructions to execute simultaneously. Most powerful microprocessors are based on many-core chips, yet most applications cannot exploit such power, requiring parallelized algorithms (Sergio Gálvez *et al.*2010) which can detect the total numbers of CPU cores and divide the task into number of subtasks as of number of cores**.** To develop such multi-core aware software we need multi-core aware libraries.

Major Bioinformatics software are developed now a days are multi-core enabled like molecular docking (O. Trott *et al.* 2010) and simulation studies **(**Sadaf R. Alam *et al.* 2008**)**. However there is no multi-core aware libraries, hence programmers take lots of pain to develop such applications. Multi-core enabled applications in the field of bioinformatics will not only reduce the time of processing but also it will consume less energy.

In the field of Bioinformatics the most widely used Java Library is Bio Java(Prlic, A *et al*,2012) and some of their modules like disorder prediction are now being made multi core enabled. In our study we have tried to develop multi-core aware java library for peptide properties calculations which are used in various active site prediction, protein stability calculations, protein-ligand interactions etc. The library is initially developed using 8 methods related to protein stability parameters. The methods are tested for various granularity of tasks ranging from .1MB to 51.2 MB protein sequence data and their processing time (in *ms*) and energy consumption (in watts) is measured and compared with corresponding Bio Java library.

The developed library (named as pBio) shows improved performance up to 37 times than corresponding modules of Bio Java, and up to 6 times more energy efficient than corresponding modules of Bio Java. Development of such multi-core aware libraries in bioinformatics software development domain is a promising field in nascent stage and availability of such library will hide the complexities of thread management lead the developers to develop multi-core aware software with reliability and accuracy.

2. Materials and Method

2.1. Material

a) Intel Core i3-2330M Processor 2.20GHz b) Open Hardware Monitor c) JDK 1.8 d) BioJava 3.0.5 e) Data : Protein sequence in FASTA file format.

2.2 Method

The methods of the library creates n (cores) number of threads which are not affined to cores for better performance (Subrata Sinha *et al*. 2010). Hence the threads created are given freedom to move from one core to another core as per the directives of OS. number of cores has been detected using ***Runtime*** class and based on the number of cores the input sequence has been divided into number of subtasks as per the number of cores detected at runtime. Each subtask has been executed using ***Fork-Join model*** of thread (KL Nitin *et al.* , 2012) The algorithm of the methods are given below

2.2.1 The instability index (II) for a protein computed using the DIWV by equation(Kunchur G *et al*.1992) as $\mathrm{II} = (10/\mathrm{L}) \sum_{i=1}^{L-1} \mathrm{DIWV}(\mathrm{xiyi}+1)$ where xiyi+1 is a dipeptide, L is the length of the sequence and 10 is a scaling factor.

2.2.2 Hydropathy Index is calculated based on Kyte and Doolittle Scale (Kyte, J *et al.*1982) by following formula: $Hydropathy\ Index = \sum_{i=1}^{n} f_i H_i$ Where f_i = frequency of amino acid i and H_i=hydropathy value of hydrophobic amino acid *i* .

2.2.3 Henderson-Hasselbach equation to calculate protein charge in certain pH. For -ve charged macromole cules pI is calculated as pI (-ve) = $\sum_{i=1}^{n} \frac{-1}{1+10^{pK_n - pH}}$ and For positive charged macromolecules pI is calculat ed as pI (-ve) pI (+ve) = $\sum_{i=1}^{n} \frac{-1}{1+10^{pK_n - pH}}$, where pKn and pKp is the acid dissociation constant of negati vely and positively charged amino acid respectively (A.Lehninger, 2005).

2.2.4 The aliphatic index of a protein is calculated according to the following formula as Aliphatic index = X(Ala) + a * X(Val) + b * (X(Ile) + X(Leu)) , where X(Ala), X(Val), X(Ile), and X(Leu) are mole percen t (100 X m ole fraction) of alanine, valine, isoleucine, and leucine and The coefficients 'a' and 'b' are the r elative volume of valine side chain (a = 2.9) and of Leu/Ile side chains (b = 3.9) to the side chain of alanine (Ikai, A.J. ,1980).

2.2.5 The extinction coefficient (Cysteine Reduced) of the native protein in water can be computed using the following equation: $E(prot) = \sum Y * ext(Y) + \sum W * ext(Y)$ where Ext(y)=1490,Ext(w)=5500 (Gill, S.C. *et al.* 1989)

2.2.6 The net charge *Z* of a peptide at a certain pH can be estimated by $Z = \sum_i N_i \frac{10^{pKa_i}}{10^{pH}+10^{pKa_i}} - \sum_j N_j \frac{10^{pH}}{10^{pH}+10^{pKa_j}}$, where *Z=Net charge of the peptide sequence*, N_i =Number of Arginine, Lysine, Histidine residue and the N-terminus, $P^{kai}=P^{ka}$ values of the N-terminus and Arginine, Lysine, Histidine residues and N_j=Number of Asperagine, Glutamic Acid, Cystiene and Tyrosine residues and the C-terminus and $P^{kaj}=P^{ka}$ values of the C-terminus and Asperagine, Glutamic Acid, Cystiene and Tyrosine residues

2.2.7 The absorbance (optical density) can be calculated using the following formula as $asb(prot) = \sum Y * ext(Y) + \sum W * ext(Y) / \sum_{i=1}^{n} Mi$, where $\sum Y * ext(Y) + \sum W * ext(Y)$ = Extinction coefficient of peptide sequence and Mi=Isotropic mass of amino acid i (Gill, S.C. *et al.* 1989)

2.2.8 Molecular Weight of Corresponding Amino Acids derived from Mascot Database.

Measurement of Energy Consumption

For measurement of power consumed by each method of the library for various granularity of tasks, we have used Open Hardware Monitor generates a log file having vital information of power consumed by CPU, CPU load, Memory usage, temperature of CPU etc at an interval of 1 second(OHM, 2015)

3. Results

Performance Enhancement Study

Methods getpInstabilityIndex(), getpMolecularWeight(), getpExtictionCoefficient(), getpAbsorbance(), getpHydropathyIndex() , getpAliphaticIndex() , getpNetCharge(), getpPI() , written in class pPeptideProperties of java was executed on Dual core(2 cores),and Quad core(4 cores) CPUs for Input sequence of various sizes (granularity) and the execution time was measured in milliseconds.

Table 3.1 Execution Time in Milliseconds of methods of Newly Developed Multi Core Aware Library pBio on Dual and Quad Core Machine for various granularity of tasks

Granularity of task(in kb)	Instability		Mw		Extinction Coefficient		Absorbance		Hydropathy Index		Aliphatic Index		Netcharge		Isoelectric Point(pI)	
	Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core	
	2	4	2	4	2	4	2	4	2	4	2	4	2	4	2	4
100	38	23	56	34	13	8	18	11	16	10	25	15	21	13	23	14
200	39	23	22	13	19	11	17	10	17	10	17	10	23	14	31	19
400	54	32	92	55	28	17	26	16	28	17	19	11	24	14	24	14
800	60	36	105	63	30	18	43	26	44	26	24	14	30	18	37	22
1600	99	59	120	72	36	22	50	30	54	32	27	16	38	23	48	29
3200	108	65	136	82	40	24	70	42	77	46	34	20	48	29	68	41
6400	120	72	150	90	45	27	110	66	99	59	58	35	55	33	85	51
12800	250	150	200	120	56	34	177	106	163	98	70	42	122	73	142	85
25600	301	181	270	162	64	38	200	120	220	132	98	59	270	162	307	184
51200	400	240	330	198	75	45	250	150	333	200	140	84	443	266	493	296

Table 3.2 Execution Time in Milliseconds of methods of Newly Developed Multi Core Aware Library pBio on Dual and Quad Core Machine for various granularity of tasks

Granularity of task(in kb)	Instability		Mw		Extinction Coefficient		Absorbance		Hydropathy Index		Aliphatic Index		Netcharge		Isoelectric Point(pI)	
	Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core		Numbers of Core	
	2	4	2	4	2	4	2	4	2	4	2	4	2	4	2	4
100	353	353	47	47	18	18	106	106	122	122	235	235	285	285	290	290
200	422	422	66	66	32	32	169	169	170	170	238	238	389	389	392	392
400	437	437	123	123	119	119	357	357	279	279	363	363	516	516	536	536
800	459	459	150	150	218	218	362	362	312	312	429	429	545	545	555	555
1600	480	480	192	192	253	253	715	715	601	601	795	795	738	738	738	738
3200	505	505	433	433	300	300	820	820	810	810	870	870	870	870	890	890
6400	690	690	470	470	350	350	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
12800	∞	∞	∞	∞	429	429	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
25600	∞	∞	∞	∞	500	500	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
51200	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

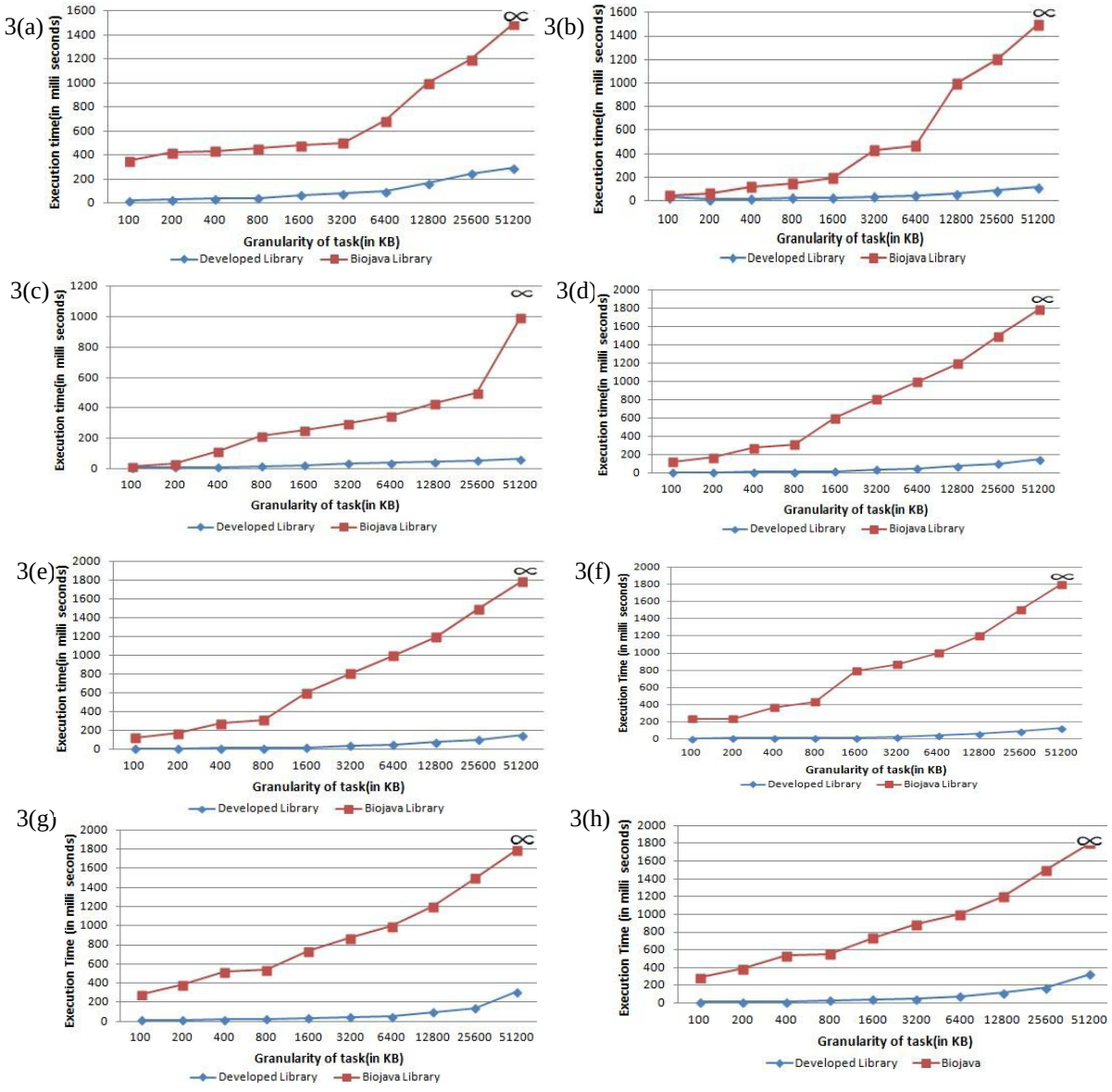


Fig 3.1 Graphical representation of Execution Time(in ms) Vs Granularity of task (in KB) for 3.1(a). Instability Index 3.1(b). Molecular Weight 3.1(c).Extinction Coefficient 3.1(d). Absorbance 3.1(e). Hydropathy 3.1(f). Aliphatic Index 3.1(g). Netcharge 3.1(h). Isoelectric Point

From table 3.1 and 3.2 it is evident that the developed library time of consumption decreases along with increase in number of cores for every granularity of task. Whereas Bio Java library gives same processing time no matter how many cores it runs on. The comparative graphical representation is given in Fig 3.1(a) to 3.1(h).

Table 3.3 Performance enhancement of multi core aware pBio library with respect to Bio Java library

Granularity of task (in kb)	Instability Index		Molecular Weight		Extinction Coefficient		Absorbance		Hydropathy Index		Aliphatic Index		Net charge		Isoelectric Point(pI)	
	Nos. of Core		Nos. of Core		Nos. of Core		Nos. of Core		Nos. of Core		Nos. of Core		Nos. of Core		Nos. of Core	
	2	4	2	4	2	4	2	4	2	4	2	4	2	4	2	4
100	91	15	1	1	1	2	6	10	8	13	9	4	14	23	13	21
200	1	18	3	5	2	3	10	17	10	17	14	16	17	28	13	21
400	8	13	1	2	4	7	14	23	10	17	19	23	22	36	22	37
800	8	13	1	2	7	12	8	14	7	12	18	32	18	30	15	25
1600	5	8	2	3	7	12	14	24	11	19	29	30	19	32	15	26
3200	5	8	3	5	8	13	12	20	11	18	26	49	18	30	13	22

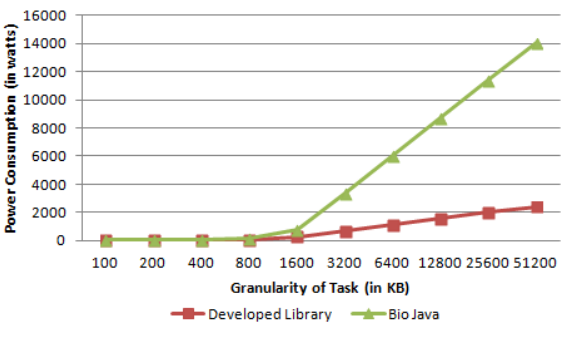
Energy Efficiency Study

Methods getpInstabilityIndex(), getpMolecularWeight(), getpExtictionCoefficient() , getpAbsorbance(), getpHydropathyIndex() , getpPI() , written in class pPeptideProperties of java was executed on Dual core(2 cores),and Quad core(4 cores) CPUs for Input sequence of various sizes (granularity) and the power consumption was measured in watts using Open Hardware Monitor.

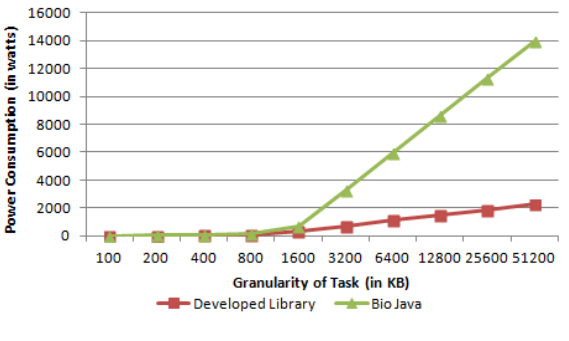
Table 3.4. Power Consumption of pBio and Bio Java library in Watts

Granularity of Tasks(in KB)	Instability Index Calculation		Molecular Weight		Extinction Cofficient		Absorbance		Hydropathy Index		Isoelectric Point	
	pBio	Bio Java	pBio	Bio Java	pBio	Bio Java	pBio	Bio Java	pBio	Bio Java	pBio	Bio Java
100	9.0	32.8	7.6	14.8	8.3	20.5	8.9	21.0	8.7	22.1	8.6	17.3
200	16.3	41.3	16.4	33.2	22.7	28.5	18.0	28.6	22.4	29.1	17.4	38.3
400	48.0	56.4	43.2	70.0	32.1	60.6	34.5	51.0	51.0	67.6	48.3	73.5
800	60.9	171.8	46.1	149.8	47.2	152.4	45.1	150.0	89.8	150.5	56.1	178.8
1600	277.2	745.9	349.8	655.3	289.1	702.5	321.5	756.0	124.4	161.6	378.8	690.2
3200	704.8	3402.7	732.1	3312.8	701.8	3452.7	600.3	3136.5	213.9	679.4	735.3	3332.8
6400	1132.4	6059.5	1114.5	5970.4	1114.5	6202.9	879.1	5517.0	303.4	1197.3	1091.9	5975.4
12800	1560.0	8716.3	1496.9	8628.0	1527.2	8953.1	1157.8	7897.6	392.8	1715.1	1448.5	8618.0
25600	1987.6	11373.1	1879.3	11285.6	1939.8	11703.3	1436.6	10278.1	482.3	2233.0	1805.1	11260.6
51200	2415.1	14029.9	2261.7	13943.2	2352.5	14453.5	1715.3	12658.7	571.8	2750.8	2161.7	13903.2

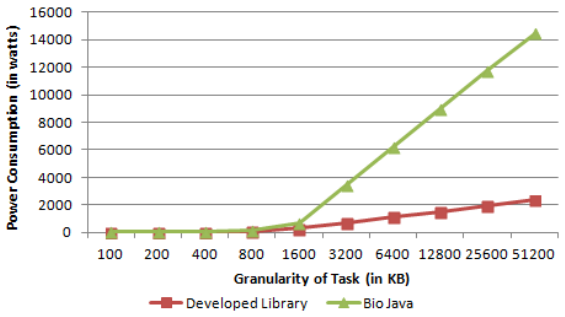
3.2 (a)



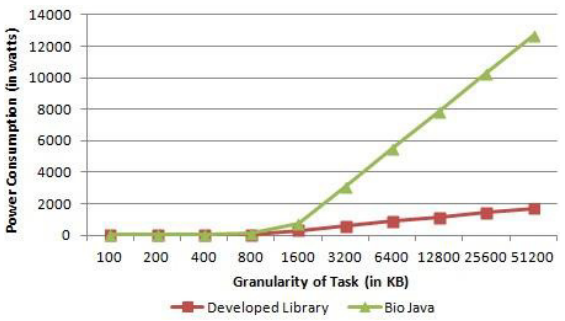
3.2(b)



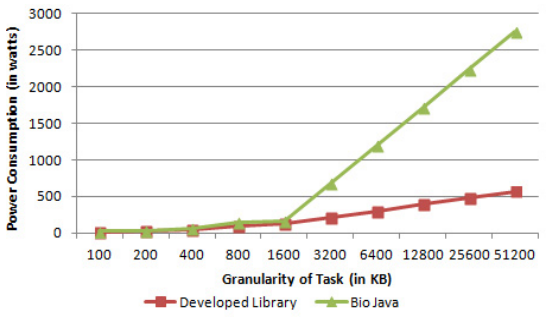
3.2(c)



3.2(d)



3.2 (e)



3.2 (f)

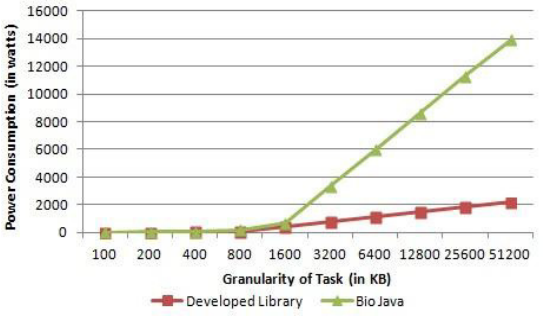


Fig 3.2 Graphical representation of Power Consumption(in watts) Vs Granularity of task (in KB) for 3.2(a). Instability Index 3.2(b). Molecular Weight 3.2(c).Extinction Coefficient 3.2(d). Absorbance 3.2(e). Hydropathy and 3.2(f). Isoelectric Point

Table 3.5. Energy Efficiency of pBio in comparison to Bio Java

Granularity of Tasks(in KB)	Instability Index Calculation	Molecular Weight	Extinction Coefficient	Absorbance	Hydropathy Index	Isoelectric Point
100	4	2	2	2	3	2
200	3	2	1	1	1	2
400	1	2	2	1	1	2
800	3	3	3	2	2	3
1600	3	2	2	1	1	2
3200	5	5	5	3	3	5
6400	5	5	6	4	4	5
12800	6	6	6	4	4	6
25600	6	6	6	5	5	6
51200	6	6	6	5	5	6

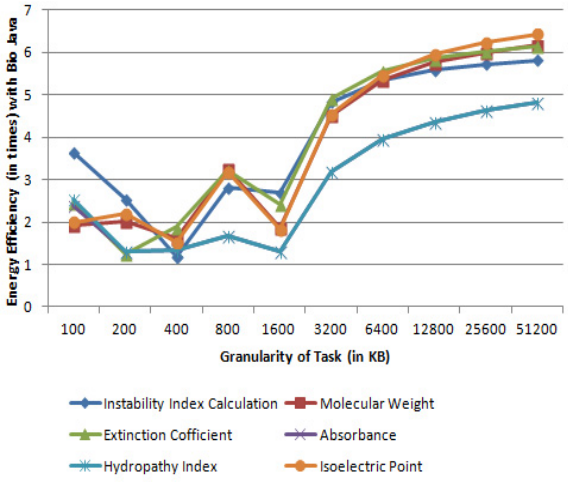


Fig 3.3 Energy Efficiency of pBio in comparison to Bio Java

The power consumption by CPU has been observed to be quite less in case of the pBio library. From table 3.4 and 3.5 and Fig 3.2(a) to 3.2(f) it can be observed that newly developed library much more energy efficient than Bio Java library. It is found to be 6 times energy efficient Instability Index, 6. times energy efficient Molecular Weight , 6 times energy efficient Extinction Coefficient, 5 times energy efficient Absorbance, 5 times energy efficient Hydropathy and 6 times energy efficient Isoelectric Point methods.

Conclusion

Development of multi-core aware libraries in bioinformatics software development domain is a promising field and availability of such library will hide the complexities of thread management lead the developers to develop multi-core aware software with reliability and accuracy.

Acknowledgements

We acknowledge Prof. Subrata Chakraborty, Director i/c Centre for Bioinformatics Studies for facilitating with the infrastructure needed to perform the experiments and we acknowledge Dr. K Narain, Dy. Director of RMRC-ICMR, Dibrugarh and Dr. Ashwani Sharma, President of Indian Science and Technology Foundation, Delhi for their constant help and support.

References

1. A.Lehninger, *Principles of Biochemistry*, 4th Edition (2005), Chapter 3, page78, Table 3-1
2. O. Trott, A. J. Olson. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading; *Journal of Computational Chemistry* 31 (2010) 455-461
3. Sadaf R. Alam,Pratul K. Agarwal,Scott S. Hampton, Hong Ong; Experimental Evaluation of Molecular Dynamics Simulations on Multi-core Systems*; Lecture Notes in Computer Science* Volume 5374, 2008, pp 131-141
4. Gill, S.C. and von Hippel, P.H. (1989) Calculation of protein extinction coefficients from amino acid sequence data. Anal. Biochem. 182:319-326(1989). [PubMed: 2610349]
5. Guruprasad, K., Reddy, B.V.B. and Pandit, M.W. (1990) Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. Protein Eng. 4,155-161.
6. Ikai, A. J. (1980) Thermostability and aliphatic index of globular proteins. J. Biochem. 88,1895–1898.
7. Kyte, J. and Doolittle, R.F. (1982) A simple method for displaying the hydropathic character of a protein. J. Mol. Biol. 157, 105-132.
8. Amino acid reference data [Internet] Available from http://www.matrixscience.com/help/aa_help.html Cited on 4th July, 2015
9. Michael Möller, Open Hardware Monitor [Internet] Available from http://openhardwaremonitor.org/documentation/ cited on 21st May, 2015.
10. Prlic, A. Yates, S.E. Bliven, P.W. Rose, J. Jacobsen, P.V. Troshin, M. Chapman, J. Gao, C.H. Koh, S. Foisy, R. Holland, G. Rimsa, M.L. Heuer, H. Brandstatter-Muller, P.E. Bourne, S. Willis, BioJava: an open-source framework for bioinformatics in 2012, Bioinformatics 28 (20) (Oct 15 2012) 2693–2695.
11. Held, J., Bautista, J., and Koehl,S., " From a Few Cores to Many: A Tera-scale Computing Research Overview", White Paper: Research at Intel.Intel Leap ahead, 2006.
12. John Darlinton, Moustafa Ghanem, Yike Guo, Hing Wing To, "Guided Resource Organisation in Heterogeneous Parallel Computing", Journal of High Performance Computing.1996
13. Sergio Gálvez,, David Díaz, Pilar Hernández, Francisco J. Esteban, Juan A. Caballero and Gabriel Dorado; Next-generation bioinformatics: using many-core processor architecture to develop a web service for sequence alignment; 2010 Vol. 26 (5): 683–686
14. Subrata Sinha, Subrata Chakraborty, M.P Barman, Aryabartta Sahu (2010) "A Study on the performance of serial code on Shared Memory Parallel Architecture", Proceedings of IEEE International Conference, Oct 28-30, 2010, ISBN: 978-1-4244-7672-5,p 19-22
15. KL Nitin, Sangeetha S ; Java 7 Fork/Join Framework [Internet] Available from http://www.developer.com/java/java-7-forkjoin-framework.html Updated on July 30, 2012 cited on 4th July 2015